Deep Learning vs. Traditional Solutions for Group Trajectory Outliers

Asma Belhadi [1], Youcef Djenouri[2], Djamel Djenouri[3], Tomasz Michalak[4], Jerry Chun-Wei Lin[5]
[1]Dept. of Technology, Kristiania University College, Oslo, Norway
[2]Dept. of Mathematics and Cybernetics, SINTEF Digital, Oslo, Norway
[3]Computer Science Research Centre, Department of Computer Science & Creative Technologies, University of the West of England, Bristol, UK
[4]Dept. of Computer Science, Warsaw University, Warsaw, Poland
[5]Dept. of Computing, Mathematics, and Physics, Western Norway University of Applied Sciences, Bergen, Norway

***Abstract*—This paper introduces a new model to identify group of trajectory outliers from a large trajectory data base and proposes several algorithms. These can be split into three categories. 1) Algorithms based on data mining and knowledge discovery, which study the different correlation among the trajectory data and identify the group of abnormal trajectories from the knowledge extracted. 2) Algorithms based on machine learning and computational intelligence methods, which use the ensemble learning and metaheuristics to find the group of trajectory outliers. 3) An algorithm exploring convolution deep neural network that learns the different features of historical data to determine the group of trajectory outliers. Experiments on different trajectory databases have been carried out to investigate the proposed algorithms. The results show that the deep learning solution outperforms the data mining, the machine learning, and computational intelligence solutions, as well as the state-of-the-art solutions in terms of runtime and accuracy performance.**

***Index Terms*—Deep Learning, Machine Learning, Computational Intelligence, Data Mining. Trajectory Data.**

I. Introduction

The problem of outlier detection involves detecting and, when appropriate, removing anomalous observations from data. This problem emerges in numerous applications [1]–[3]. A general form of this problem is *group outlier* (or *anomaly*) *detection* in which the aim is to identify sets of anomalous observations rather than only individual ones [4], [5]. Sample applications include locating unusual clusters of celestial objects from image processing [5], astronomical data [6], machine monitoring [7], article and website management [8]. In this paper we consider *trajectory outlier detection*—a variant of the outlier detection problem for spatial data that involves trajectories. This includes, for instance, vehicle positioning data, hurricane tracking data, and animal movement data. The aim is to identify trajectories that deviate from regular paterns. [9]. This domain has been boosted by the proliferation of GPS-enabled devices that produce countless trajectories. The identification of outliers in such data is important to optimize routes in the short term, e.g. in car navigation systems, or to make longer-term decisions, such as improving the organization of an urban area [10].

Unlike the general outlier detection problem, prior works on the trajectory outlier detection problem consider solely *individual* outliers (see the next section for an overview). However, in real-world scenarios, trajectory outliers can appear in groups, e.g., a group of vehicles that deviates from an usual trajectory due to the maintenance of streets in the context of intelligent transportation, or a group of hurricane trajectories that deviates from the normal hurricane ones in the context of climate change [11]. We introduce and analyse in this paper the *Group Trajectory Outlier detection* (GTO) problem and explore different solutions based on clustering, neighborhood computation, feature selection, and ensemble learning. High performance computing (HPC) is explored and a GPU-based solution is proposed to deal with big trajectory databases in a fast way. The main contributions in this paper can be summarized as follows:

- The group trajectory outlier detection problem is formally defined by considering the individual trajectory outliers as trajectory candidates and introducing a new concept of the density of a group of trajectory outliers. A group of trajectory outliers is thus defined as a set of individual trajectory outliers that are highly correlated, i.e., with a high number of shared locations.
- The *DBSCAN* algorithm [12] and kNN [13] are revisited and adapted for the group trajectory outlier (GTO). This results in *DBSCAN-GTO*, and kNN-GTO, respectively. DBSCAN-GTO starts by applying the *DBSCAN* method to derive the *micro clusters*. These micro clusters are then considered as potential candidates for which we propose a pruning strategy based on density computation measure. The pruning produces groups of trajectory outliers. kNN-GTO starts by recursively deriving the trajectory candidates from the individual trajectory outliers and then prunes these candidates using the proposed density computation measure.
- Three more advanced algorithms are developed. The *FS-GTO* algorithm is first proposed, which models the group trajectory outlier detection problem as feature selection problem. In particular, the set of individual trajectory

outliers are considered as the set of all features, and the feature selection process is adopted to identify the group of trajectory outliers. A computational intelligence model is then proposed by considering swarm intelligence behaviors in exploring the solution space of group of trajectory outliers. A deep learning model is finally proposed by considering the GTO problem as an object detection problem. The image trajectories database is first collected, then the convolution neural network is applied to detect and identify the group of trajectory outliers.

- The performance of the proposed algorithms is evaluated using different real trajectory databases. Since, to the best of our knowledge, this is the first work that explores group trajectory outlier detection, we compare the proposed solutions with general group outlier detection solutions (see Section II for an overview). The results demonstrate that the proposed algorithms outperform the baseline algorithms for group detection. Furthermore, the experiments show the scalability of the three approaches and the ability of HPC approach to efficiently deal with large trajectory databases.

The remainder of the paper is organized as follows. Section II reviews the main existing trajectory and group outlier detection algorithms. We formally define the problem in Section III. Section IV presents the proposed algorithms for tackling the GTO problem, namely, *DBSCAN-GTOD*, kNN-GTOD and *FS-GTOD*. Section VIII presents the performance evaluation of our proposed algorithms. Discussion of the the learned lessons, conclusions, and directions for future work are given in Section IX.

II. Related Work

Solutions for trajectory outlier detection are distance-based fall into three main categories, 1) distance-based [14]–[19], 2) density-based [20]–[26], and 3) pattern mining based [27]–[32]. There are a few further works that are based on machine learning [33]–[37].

The first category is based on neighborhood computation, while the density-based approaches aim to compute the density of each trajectories and consider trajectories with low density values as outliers. Pattern mining-based approaches explore the different correlations among trajectories to find the outliers. Approaches based on machine learning learn the outlier detection process from the training trajectories to identify anomalies in the new inserted trajectories. All the existing solutions focus on discovering individual outliers, whereases trajectory outliers appear in groups in real life scenarios. Group outlier detection (in its general context) have been largely studied in the last decade, and the solutions proposed the literature may be classified into three main categories:

1) Statistical models: Chalapathy et al. [5] consider the use of a deep generative model and test it on various image applications. The outlierness for each group in the input data is estimated by group reference function using a standard back-propogation algorithm. Liang et al. [6] use a topic modeling-based approach to find group outliers. The inference is performed by gibbs sampling, and the learning is done by monte carlo algorithm. Das et al. [38] considers the different correlation between the data outliers to detect pattern anomalous by investigating bayesian network anomaly detection, and conditional anomaly detection. Thus, the correlation score between the individual outliers is determined by the probability of possible values of these outliers in the training data.
2) Contextual models: Tang et al. [39] defines contextual outlier detection as small group of points that share similarity, on some attributes, with a significantly larger reference group of data, but deviates dramatically on some other attributes. In order to avoid enumerating all contextual outliers, they only maintain the closure context outliers. In addition, only contextual outliers with a statistical significance test greater than a given threshold are retrieved. Li et al. [40] assigns feature weights on each group outlier, and compute chain rule entropy to determine correlation between different feature groups. Zhao et al. [41] design a parallel computing solution to deal with contextual outlier detection in high and sparse dimensional space. Xiong et al. [42] studied detecting two kinds of group anomalies: a group of individual anomalous points, and a set of normal points the distribution of whom as a group is abnormal. The authors define a mixture of gaussian mixture model by adopting the likelihood of each group, the marginal likelihood of each observation within a group, and the maximum likelihood estimation to learn the hyperparameters of the mixture model. An application of this algorithm in social media analysis is investigated in [43] by taking into account the dynamic properties of the social media data.
3) Clustering models: These approaches use clustering strategies on the individual of outliers to group these outliers into similar clusters [38], [39], [44]. Each cluster is then considered as a group of outliers. Soleimani et al. [44] proposed supervised learning approach that groups anomalous patterns when memberships are previously unknown. The salient features are extracted from an appropriate training set with discrete data inputs. It implements a nonparametric bootstrap sampling procedure to evaluate the statistical significance of a detected anomalous behavior for a single object as well as a cluster of objects. The approach is applied on topic documents modeling and it is able to discover irregular topic mixtures from a collection of documents. Sun et al. [45] proposed abnormal group-based joint medical fraud approach. The abnormal group problem is converted to the maximal clique enumeration problem [46] by considering the set of patients as the set of vertices, and each edge indicates that the two connected patients are similar. Note that, the similarity between patients is determined by computing their identical joint behaviors. Maximal clique enumeration is NP-hard problem, to do such task efficiently different partition strategies [47] are investigated to reduce the graph size. As a result, each maximal clique is considered as abnormal group of patients.

These algorithms they are not dedicated to trajectory data. They focus on finding a group outliers from the set of candidate groups, and not from the individual outliers. We propose herein the first algorithms that detects group *trajectory* outliers from individual trajectory outliers.

III. Definitions and Problem Statement

A few preliminary definitions are needed before introducing the group trajectory outlier problem. A trajectory is a sequence of location points in space. We will denote by pt a single spatial location point, where each pt is a tuple of two values— the latitude and the longitude of this location.

Definition 3.1 (Trajectory Database): We define a trajectory database $T = \{T_1, T_2...T_m\}$, where each raw trajectory T_i is a sequence of spatial location points $(p_{i1}, p_{i2}...p_{in})$, obtained by localization techniques such as GPS. Each point is represented by the latitude, and the longitude values, respectively.

As common in the literature [48], the location points which are similar enough are aggregated into regions. Let us denote by R a location *region* in space.

Definition 3.2 (Mapped Trajectory Database): We define a mapped trajectory database $\Lambda = \{\Lambda_1, \Lambda_2...\Lambda_m\}$, where each mapped trajectory Λ_i is a sequence of spatial location regions $(R_{i1}, R_{i2}...R_{in})$, obtained by mapping each point in T_i to the closest region R_i. We note $R = \{R_1, R_2...R_{|R|}\}$, by the set of all regions.

We define the dissimilarity between any two trajectories as the distance between them.

Definition 3.3 (Trajectory Dissimilarity): We define the distance between two trajectories $d(\Lambda_i, \Lambda_j)$ by the number of all regions minus the number of shared regions between the two trajectories Λ_i, and Λ_j, as

$$d(\Lambda_i, \Lambda_j) = max_{ij} - inter_{ij} \tag{1}$$

where,

$$inter_{ij} = |\{(R_{il}, R_{jl}) | R_{il} = R_{jl}, \forall l \in [1..n]\}| \tag{2}$$

and,

$$max_{ij} = max(|\Lambda_i|, |\Lambda_j|) \tag{3}$$

and,

$|\Lambda_i|, |\Lambda_i|$ are the number of regions of trajectories Λ_i, Λ_j, respectively.

We define the groups of trajectory candidates, i.e., the set of potential trajectories belong to the groups of trajectory outliers. These trajectory candidates are retrieved from the individual trajectory outliers.

Definition 3.4 (Group Trajectory Candidate): We define a group of trajectory candidate $\mathcal{G}$ by the set of individual trajectory outliers retrieved from the set of individual trajectory outliers ITO, i.e.,

$$\mathcal{G} = \{\Lambda_i | \Lambda_i \in ITO\} \tag{4}$$

Note that ITO could be retrieved using one of the well known trajectory outlier detection algorithms such as the local outlier factor, and the k nearest neighbors.

The density of a group is an important concept in our analysis. Intuitively, it is defined as the ratio between the number of trajectories of the group and the number of shared regions among the trajectories of such group.

Definition 3.5 (Density Group): We define the density of the candidate group trajectory outliers $\mathcal{G}$ as

$$Density(\mathcal{G}) = \frac{|\mathcal{G}|}{|\{R_j | \Lambda_i \in \mathcal{G}, R_j \in \Lambda_i\}|} \tag{5}$$

To normalize the density function, we divide the result by the density of the group having maximum density value, this ensures to obtain values ranged from 0 to 1. We call this function *N*ormalizedDensity.

Furthermore, we formally define the concept of a group of trajectory outliers.

Definition 3.6 (Group Trajectory Outlier): A set of trajectories $\mathcal{G}$ is called a Group Trajectory Outlier if and only if,

$$\begin{cases} \mathcal{G} \subseteq ITO \\ NormalizedDensity(\mathcal{G}) \geq \gamma \end{cases} \tag{6}$$

Note that γ is the density threshold varied from $[0 \dots 1]$.

Group trajectory outliers may be redundant, we define the non-redundant group trajectory outliers, as follows,

Definition 3.7 (Non-Redundant Group Trajectory Outlier): A group of trajectory outliers $\mathcal{G}$ is called a Non-Redundant Group Trajectory Outlier if it has no superset of $\mathcal{G}$, that is a group of trajectory outlier.

Now, we are ready to formally define the group trajectory outlier problem.

Definition 3.8 (Group Trajectory Outlier Problem): Group Trajectory Outlier Problem aims to discover from the set of all mapped trajectories, the set of all non-redundant groups of trajectory outliers, denoted by $\mathcal{G}^*$.

Trivial approach for solving the group of trajectory outlier detection consider all possible combinations between the mapped trajectories, and evaluates each subset separately using Def. 3.5. The closed group of trajectory outliers are then derived. This method requires high computational and memory resources, to evaluate the candidate sets and save the potential groups of trajectory outliers. The theoretical complexity of this approach is $O(2^{|\Lambda|})$. To address these issues, we propose in the next section alternative framework to improve the group trajectory outlier detection process.

IV. General GTO Framework

As shown in Fig. 1, the proposed framework includes:

1. **Mapping (Pre-processing):** Typically, trajectories in most applications consist of noisy GPS data points where errors can exceed several meters. This can negatively influence the final output of many algorithms. Hence, a map-matching step should first be used to project GPS data points of each trajectory onto a road network. Several approaches have been developed to achieve this [49]–[51]. Since in this work we are interested in sparse trajectory databases, we use the probabilistic model based on a Hidden Markov Model [50], [52]. In this model, each road segment is represented as hidden state in the Markov chain— with an emission probability representing the likelihood of observing the GPS point conditional on the candidate road segment being the true match. A

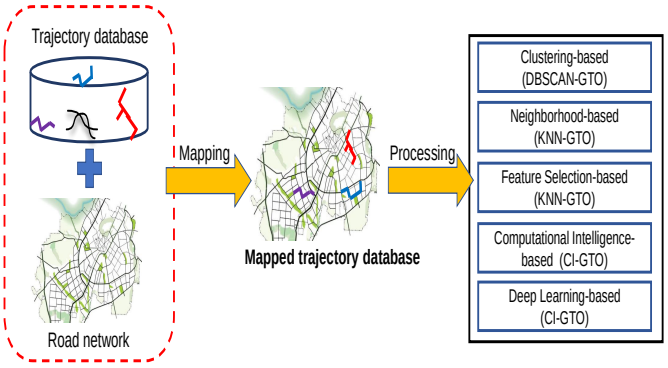

Fig. 1. Data Mining-based Solutions

higher probability to a road segment is assigned if the observed trajectory points are close to it. The maximum likelihood path over the Markov chain that has the highest probability is then determined, and the corresponding road segment is associated to the observed trajectory point. This way, the mapped trajectory database is created and every observed trajectory is assigned to the associated road segment.

2. **Processing:** After constructing the mapped trajectory database, a processing step is performed to find out the group of trajectory outliers. In this context, two methods for finding group of trajectory outliers are proposed. A few approaches are investigated including clustering, neighborhood computation, and feature selection. The first method is to start by determining the individual trajectory outliers and then find out the group of trajectory outliers. The second method is to derive directly the group of trajectory outliers from the mapped trajectory database. Furthermore, we present various approaches for improving the performance of our techniques by incorporating ensemble learning and HPC.

V. Data Mining-based Solutions

This approach builds upon data mining techniques including clustering, feature selection, neighborhood computation to develop the dedicated algorithms. In the remainder of this section, we describe the details of the proposed algorithms (Sections V-A, V-B, and V-C).

A. The DBSCAN-GTO Algorithm

To present the adaptation of the DBSCAN algorithm [12], we need the following three definitions:

Definition 5.1 (Trajectory Neighborhoods): We define the neighborhoods of a trajectory Λ_i, $\mathcal{N}_{\Lambda_i}$, for a given threshold ϵ by

$$\mathcal{N}_{\Lambda_i} = \{\Lambda_j | d(\Lambda_i \Lambda_j) \leq \epsilon \vee j \neq i\}. \quad (7)$$

Definition 5.2 (Core Trajectory): A trajectory Λ_i is defined as a core trajectory if there is at least a minimum number of trajectories *MinPts* such that $|\mathcal{N}_{\Lambda_i}| \geq$ *MinPts*.

Definition 5.3 (Micro Cluster): A cluster of trajectories C_i is defined as a micro cluster if and only if $0 < |C_i| \leq \mu$, where μ is a user threshold.

Algorithm 1 DBSCAN-GTO Algorithm

Input: $\Lambda = \{\Lambda_1, \Lambda_2 ... \Lambda_n\}$: The set of all trajectories.
ϵ, *MinPts*: DBSCAN parameters.
μ: User threshold for micro clusters.
γ: density threshold.
Output: $\mathcal{G}^*$: sets of all group trajectory outliers.
C $\leftarrow$ DBSCAN(Λ, ϵ, *MinPts*)
$\mathcal{G}^* \leftarrow \emptyset$
for each $C_i \in C$ **do**
  if $|C_i| \leq \mu \vee \text{Density}(C_i) \geq \gamma$ **then**
    $\mathcal{G}^* \leftarrow \mathcal{G}^* \cup C_i$
  end if
end for
return $\mathcal{G}^*$

In general, solutions to trajectory clustering [53], [54] are able to derive clusters with different densities. However, these algorithms do not explore the *micro clusters* property for anomaly detection. This section presents our approach for identifying group of trajectory outliers, *DBSCAN-GTO*, that uses the DBSCAN algorithm to search for clusters by checking the ϵ-neighborhood of each trajectory (See Definition 5.1). The core trajectories are determined using Definition 5.2. DBSCAN-GTO then iteratively collects density-reachable trajectories from these core trajectories directly, which may involve merging a few density-reachable clusters. The process terminates when no new trajectories can be added to any cluster. Initially, the set of trajectories are grouped (as in DBSCAN). This generates several clusters with different sizes. Each micro cluster (See Definition 5.3) is considered as group candidates. For each group, the density of each group is determined using Definition 3.5, if the density exceeds γ threshold, then the group is selected as outlier. Algorithm 1 presents the pseudo-code of *DBSCAN-GTO*.

B. The kNN-GTO Algorithm

Let us begin the presentation of our adaptation of the kNN algorithm [13], with the following definition:

Definition 5.4 (kNN): We define kNN of a trajectory Λ_i, denoted by $k\text{NN}(\Lambda_i)$, as

$$k\text{NN}(\Lambda_i) = \{\Lambda_j \in \Lambda \setminus \{\Lambda_i\} | d(\Lambda_i, \Lambda_j) \leq k_{dist}(\Lambda_i)\}, \quad (8)$$

where $k_{dist}(\Lambda_i) = d(\Lambda_i, \Lambda_l)$ is the k-distance trajectory defined by the set of k trajectories $\Lambda' \in \Lambda$, such that $d(\Lambda_i, \Lambda_l) \geq d(\Lambda_i, \Lambda')$.

The following proposition holds:

Proposition 5.1: Let us consider two trajectories Λ' and Λ''. Let $\mathcal{G}^*(t)$ be a group of trajectory outliers at the iteration t such that:

$$\Lambda' \in \bigcup_{\Lambda_i^* \in \mathcal{G}^*(t)} k\text{NN}(\Lambda_i^*) \vee \Lambda'' \notin \bigcup_{\Lambda_i^* \in \mathcal{G}^*(t)} k\text{NN}(\Lambda_i^*).$$

Then, the following holds:

$$\Lambda' \notin \mathcal{G}^*(t+1) \Rightarrow \Lambda'' \notin \mathcal{G}^*(t+1).$$

Proof 5.1: We have that:

$$\Lambda' \in \bigcup_{\Lambda_i^* \in \mathcal{G}^*(t)} k\text{NN}(\Lambda_i^*) \vee \Lambda'' \notin \bigcup_{\Lambda_i^* \in \mathcal{G}^*(t)} k\text{NN}(\Lambda_i^*)$$
$$\Rightarrow Density(\mathcal{G}^*(t) \cup \{\lambda''\}) \leq Density(\mathcal{G}^*(t) \cup \{\lambda'\}) \ldots \quad (9)$$

Algorithm 2 kNN-GTO Algorithm

Input: $\Lambda = \{\Lambda_1, \Lambda_2 ... \Lambda_n\}$: The set of all trajectories.
$\mathcal{A}$: trajectory outlier detection algorithm.
γ: density threshold.
Output: $\mathcal{G}^*$: sets of all group trajectory outliers.
$\mathcal{G}^+_{\mathcal{A}} \leftarrow \mathcal{A}(\Lambda)$
for each trajectory $\Lambda^+_i \in \mathcal{G}^+_{\mathcal{A}}$ **do**
  $node \leftarrow \Lambda^+_i$
  for each trajectory $t \in$ (kNN$(node) \cap \mathcal{G}^+_{\mathcal{A}}$) **do**
    if Density$(\mathcal{G}^*_i \cup \{t\}) \geq \gamma$ **then**
      $\mathcal{G}^*_i \leftarrow \mathcal{G}^*_i \cup \{t\}$
      $\mathcal{G}^+_{\mathcal{A}} \leftarrow \mathcal{G}^+_{\mathcal{A}} \setminus \{t\}$
      {repeat lines from 5 to 8 for a trajectory t}
    end if
  end for
  if $|\mathcal{G}^*_i| = 1$ **then**
    $\mathcal{G}^* \leftarrow \mathcal{G}^* \setminus \{\mathcal{G}^*_i\}$
  end if
end for
return $\mathcal{G}^*$

$$\Lambda' \notin \mathcal{G}^*(t+1) \Rightarrow Density(\mathcal{G}^*(t) \cup \{\lambda'\}) \leq \gamma \ldots \quad (10)$$

From (9) and (10) we have: $Density(\mathcal{G}^*(t) \cup \{\lambda''\}) \leq \gamma \Rightarrow \Lambda'' \notin \mathcal{G}^*(t+1)$

It follows from the above proposition that if a trajectory Λ_i belongs to the *k nearest neighbors* of at least one trajectory in the current group of trajectory outliers, and Λ_i is not in the group of trajectory outliers of the next iteration, then, any trajectory that belongs to the *k nearest neighbors* of Λ_i could not be in the group of trajectory outliers of the next iteration. Consequently, it is judicious to prune the search into *k nearest neighbors* of the individual trajectory outliers. In particular, it considers as input the set of the first p individual trajectory outliers $\mathcal{G}^+ = \{\Lambda^+_1, \Lambda^+_2 ... \Lambda^+_p\}$, ranked according to the kNN value, i,e, $\forall i \geq j$, $k\text{NN}(\Lambda^+_i) \geq k\text{NN}(\Lambda^+_j)$. The process aims to enumerate the sets of group trajectory outliers, $\mathcal{G}^*$, by exploring a search tree of $\mathcal{G}^+$. It starts by adding the individual trajectory outlier ranked first, Λ^+_1, to the group trajectory outliers, denoted by $\mathcal{G}^*_1$. It then generates all potential candidates from Λ^+_1. A trajectory t is a potential candidate from Λ^+_1, if and only if, $t \in \mathcal{G}^+ \vee t \in k\text{NN}(\Lambda^+_1)$. The density of $\mathcal{G}^*_1$ is updated by adding the potential candidates to $\mathcal{G}^*_1$, one by one. Only the potential candidates respecting the density threshold are saved, and the remaining ones are removed. Once the potential candidate is added to $\mathcal{G}^*_1$, it is removed from $\mathcal{G}^+$. If $\mathcal{G}^*_1$ contains less than two elements, it is removed from $\mathcal{G}^*$. The same process is recursively applied to all potential candidates added to $\mathcal{G}^*_1$, and the overall process is repeated for all trajectory outliers in $\mathcal{G}^+$. Algorithm 2 presents the pseudo-code of our kNN-GTO algorithm.

C. FS-GTO

The following defins the transformation from the GTO problem to the feature selection problem.

Definition 5.5 (Transformation to FS Problem): Consider GTO problem $\langle \mathcal{G}^+_{\mathcal{A}}, \mathcal{G}^* \rangle$. It is transformed into the feature selection problem, represented by the set of all features F and the subset of selected features F^*, as follows: $F = \mathcal{G}^+_{\mathcal{A}}$, and F^*=$\mathcal{G}^*$. We evaluate F^* as follows:

$$Eval(F^*) = Quality(F^*) - \frac{|F^*|}{|F|}, \quad (11)$$

Algorithm 3 FS-GTO Algorithm

Input: $\Lambda = \{\Lambda_1, \Lambda_2 ... \Lambda_n\}$: The set of all trajectories.
$\mathcal{A}$: trajectory outlier detection algorithm.
γ: density threshold.
Output: $\mathcal{G}^*$: sets of all group trajectory outliers.
$\mathcal{G}^+_{\mathcal{A}} \leftarrow \mathcal{A}(\Lambda)$
$Ranking \leftarrow SPEC(\mathcal{G}^+_{\mathcal{A}})$
for true **do**
  $\mathcal{G}^* \leftarrow BFS(\mathcal{G}^+_{\mathcal{A}}, \gamma, Ranking)$
end for
return $\mathcal{G}^*$

where Quality(F^*) is computed as per Definition 3.5.

We consider each individual trajectory outlier as one feature, and our aim is to select the most relevant features from the set of all features. This set then becomes the group of trajectory outliers (see Definition 5.5). The evaluation of the selected set of features (trajectories) is computed using the group density measure, see eq. (11). The process starts by applying the feature selection algorithm on the set of individual trajectory outliers, The output of this step is a ranking of individual outliers in the descending order in terms of score feature relevance. A ranking vector is created, and a search enumeration tree is generated through a breadth-first-search (BFS). If the quality of the current group candidate does not reach the criteria from Definition 3.5, a backtracking procedure is launched by taking the next trajectory in the ranking vector. While exploring the enumeration tree of individual outliers, the aim is to maximize the function reported in eq. (11). Algorithm 3 presents the pseudo-code of our proposed FS-GTO algorithm.

VI. Computational Intelligence-based solution

Evolutionary computing and swarm intelligence draw ideas from natural evolution such as survival of the fittest, natural selection, reproduction, mutation, competition and symbiosis. The aim of the computational intelligence methods, such as genetic algorithms, genetic programming, mimetic algorithms, immune and swarm intelligence algorithms, is to make an accurate solution for the given optimization problems. In this work, the genetic algorithm and the particle swarm optimization are used to improve the quality of the returned group of trajectory outliers. The main components of the computational intelligence based solutions are:

1) Solution Space:This is all the possible combination of group trajectory outliers, i.e., the solution space size of m trajectories is $2^{|m|}$.
2) Population Initialization: The initial population is first generated by randomly selecting the group of trajectory outliers from the set of the mapped trajectories Λ. Each solution in the initial population will be one potential group of trajectory outliers.
3) Fitness Computing: The fitness of the solution, S, is computed by the density value of its group. The aim is to maximize this function. That is:

$$Fitness_{max}(S) = Density(S) \quad (12)$$

We present in the following two algorithms. The first one is based on genetic algorithm (GA), and the second is based on particle swarm optimization (PSO).

1) GA: After generating the initial population, two operators are used to refine the solutions: i) The first one is crossover. It takes two chromosomes from the population and generates two new chromosomes by making intersection operator between the groups selected. ii) The second is mutation. It takes two generated chromosomes on the crossover step and generates two more chromosomes by making the union operator between the groups selected. All generated chromosomes are evaluated by the fitness function using Eq. 12. The best chromosomes are selected for the next iteration. This process is repeated until a maximum number of iterations are reached.
2) PSO: After generating the initial population, two operators are used to update the positions of the particles. Consider a position vector $X_i^t = (x_{i1}x_{i2}x_{i3}\dots x_{in})^T$ and a velocity vector $V_i^t = (v_{i1}v_{i2}v_{i3}\dots v_{in})^T$ at iteration t for every particle i that composes it. The particles update their positions in the solutions using the velocity formula as follows:

$$V_i^{t+1} = w \times V_i^t + c1 \times (p^t - X_i^t) + c2 \times (p^* - X_i^t) \quad (13)$$

and

$$X_i^{t+1} = X_i^t + V_i^{t+1}, \quad (14)$$

where $i = 1, 2, \dots, P$.
From Eq. 13, it shows that two factors c_1, and c_2 contribute to the movement of a particle in an iteration. p^t is the position of the best particle at iteration t, and p^* is the position of the best particle of all iterations. Furthermore, Eq. 14 is used to update the position of a particle. The parameter w is a positive constant value. This parameter is important for balancing the global search (also known as exploration when higher values are set), and local search (known as exploitation when lower values are set). The new positions of the particles are evaluated by the fitness function using Eq. 12. The best positions are selected for the next iteration. This process is repeated until a maximum number of iterations are reached.

Moreover, we proposed the CI-GTO algorithm that uses both GA and PSO in exploring the solutions space. The best final results in GA and PSO are considered as the best returned solutions of CI-GTO.

VII. Deep Learning-Based Solution

This section presents the proposed CNN-GTO framework, which uses a convolutional neural network (CNN) for identifying group of trajectory outliers. CNN-GTO consists of two stages:

1) Data Collection: The aim of this step is to collect the trajectory data and build the images database. A visual strategy is used, in which each image contains a set of human behavior trajectories. The process starts by recording video frames from cameras. The frames are then transformed to images, and different distortion techniques such as mapping, resizing, are used to correct the images. The whole trajectory images are stored into the database for training.
2) Training: The CNN is applied to design a training model which is considered as a powerful vision machine to learn from the different features of the image trajectory data. The group of trajectory outlier detection is transformed to the object detection problem by setting the input of the object detection model to the trajectories image database, and the output of the object detection model to the group of trajectory outliers in each image. The regional convolution neural network of [55], [56] is used. For every image in the input, regions of interests are determined and passed to the hidden layer where the Relu activation function is performed. A similar process to the convolution neural network is followed. In fast RCNN, the model performs better and quicker as the regions of interest are found using a selective search method, and all the regions of interests of an image are found at once. This is different from CNN that finds ROI (regions of interest) and applies Relu on each ROI separately, which is slower. The process is repeated for a given number of epochs, or until the training stop providing improvement for a given number of iterations. The weight initialization is done using the pretrained ImageNet model[1]. The trained model is stored on the central workstation.
3) Inference: The aim is to derive the group of trajectory outliers of the input image using the trained model of the previous step. Thus, a propagation of the different weights of the trained model is performed to detect the objects of the image. The detected objects are considered as the group of trajectory outliers. In this step, different kinds of inference are generated. We send the trained model to the computers and infer the model for each new trajectories image data. We can also use smartphones which support Andorid, and GPU computing to infer the group of trajectory outliers in real time processing. In this context, several technologies could be integrated such as TensorflowLite [2].

In practice, the training images have a high resolution, from 5.000 pixels to 100.000 pixels. Consequently, millions to billions of region proposals have been generated. This makes the whole system very hungry in time-processing and memory. In some cases, the system will be bluntly blocked after several days and weeks of processing. To deal with this problem, we propose a strategy to prune and filter the number of bounding boxes. Two groups of trajectory outliers in the same frame should not be close to each other, and thus two bounding boxes in the same image should not be close to each other. The similarity degree is then computed between each new generated bounding box and the bounding boxes that have already been generated. The similarity between two bounding boxes is determined by the number of pixels that separate theme. Only the bounding boxes that gives high diversity of the image are kept. That is, the minimal set of bounding boxes that covers the maximum number of pixels in the image.

[1] http://www.image-net.org/
[2] https://www.tensorflow.org/lite

VIII. Experimental Evaluation

In this section, we evaluate experimentally the GTO framework and its different components. In particular, the serial implementations of the different GTO solutions are compared with the state-of-the art group outlier detection algorithms using standard trajectory databases. In addition, the scalability performance of the deep learning implementation is carried out on big trajectory databases.

Experimental Setup: The different components of the GTO framework have been implemented on Python. The experimental evaluation of the serial implementations has been performed on a computer with $64bit$ core $i7$ processor running Windows 10 and $16GB$ of RAM. The evaluation of the deep learning implementation has been carried out on a CPU host. In general, a common problem of outlier detection techniques is the evaluation procedure. This is particularly the case for new applications such as the GTO problem, where a ground truth is typically unknown. To facilitate a quantitative evaluation, for group trajectory outlier detection techniques, we adapt the process of Zhang et al. [57] to inject synthetic group trajectory outliers. In particular:

- **Injecting individual trajectory outliers**: Individual trajectory outliers are generated by adding noise *several times* with a certain probability $p \sim \mathcal{U}(0.0, 1.0)$ and a given threshold μ.
- **Injecting group trajectory outliers**: Group of trajectory outliers are generated by adding noise to the set of individual trajectory outliers, but now only a *few times* with a certain probability $p \sim \mathcal{U}(0.0, 1.0)$ and a given μ.

Note that $\mathcal{U}(0.0, 1.0)$ is the continuous uniform distribution with lower and upper bounds set to 0, and 1, respectively. For both injections, each point p_{il} in the trajectory Λ_i is changed related to the number of regions n as follows:

$$p_{il} = \begin{cases} p_{il} + n \sim \mathcal{N}(0,1) & \text{if } p \geq \mu \\ p_{il} & \text{otherwise.} \end{cases} \quad (15)$$

The evaluation is performed using Fmeasure, and ROCAUC, which are common measures for the evaluation of outlier detection methods, which are given by,

$$Fmeasure = \frac{2 \times Recall \times Precision}{Recall + Precision}. \quad (16)$$

$$Recall = \frac{|O_A \bigcap O|}{|O|}. \quad (17)$$

$$Precision = \frac{|O_A \bigcap O|}{|O_A|}. \quad (18)$$

$$AUC = mean_{(o,i)} \begin{cases} 1 & if\ score(o) > score(i), \\ 0.5 & if\ score(o) = score(i), \\ 0 & if\ score(o) < score(i), \end{cases} \quad (19)$$

where
O: The set of all outliers in the dataset.
O_A: The set of outliers returned by the algorithm.
I_A: The set of inliers returned by the algorithm.
o: Group of trajectory outliers in O_A.
i: Subset of trajectory inliers in I_A.

TABLE I
Parameter setting of CI-GTO

Population Size	Maximum Number Iterations	Intelligent Transportation	Climate Change	Environment
20	20	0.75	0.72	0.71
	50	0.76	0.73	0.71
	100	0.77	0.74	0.72
50	20	0.76	0.72	0.72
	50	0.77	0.73	0.72
	100	0.78	0.75	0.73
100	20	0.77	0.73	0.72
	50	0.78	0.74	0.73
	100	0.78	0.75	0.73

Data Description: In our experimental evaluation, we used well-known trajectory databases from different domains, i.e.:

1) **Intelligent Transportation:** We used a database from the *ECML PKDD 2015* databases competition[3]. The database contains real trajectories retrieved from $01/07/2013$ to $30/06/2014$ of *442* taxis in the city of Porto, in Portugal. This allows to recuperate more than *3 GB* of data stored in one single CSV file. Each row contains information related to one trip including: *TripID*, *CallType* and *TaxiID*. The last component of the row contains a list of GPS coordinates. This list contains one pair of coordinates for each *15* seconds of trip. The last list item corresponds to the trip's destination while the first one represents its start.
2) **Climate Change:** We used the *hurricane track data set* which contains latitude, longitude, maximum sustained surface wind, and minimum sea-level pressure of hurricane trajectories in USA at 6 hourly intervals. We use the Atlantic hurricanes [11] retrieved from the years 1851 to 2018. This data contains 52775 hurricane trajectories.
3) **Environment:** We use a database of the *Starkey Project*[4]. We consider the animal movement data illustrated by the radio-telemetry locations of *elk*, *deer*, and *cattle* retrieved from 1989 to 1999. The locations are recorded at 30 minute intervals. This data is considered sparse one with 100 trajectories, and more than 40,000 different points.

In addition, we used two additional big databases in our experiments: i) *taxi 13-1* containing 1.89 million trajectories, and ii) *taxi 13-2* containing 3.69 million trajectories [9]. In the remainder of this section, we present the results of the experiments.

A. Parameters Settings of GTO Framework

The first part of this experiment focuses on tuning the parameters of different proposed GTO solutions. As shown in Fig. 2, Tab. I, and Tab. II, several tests have been performed by varying the user threshold (from 1 to 10) for DBSCAN-GTO, the number of neighborhood (from 1 to 10) for kNN-GTO, and the tree depth (from 1 to 10) for FS-GTO, population size, and maximum number of iterations (from 20 to 100) for CI-GTO, different CNN architectures (AlexNet, VGG16, and VGG19),

[3]http://www.geolink.pt/ecmlpkdd2015-challenge/dataset.html
[4]https://www.fs.fed.us/pnw/starkey/introduction.shtml

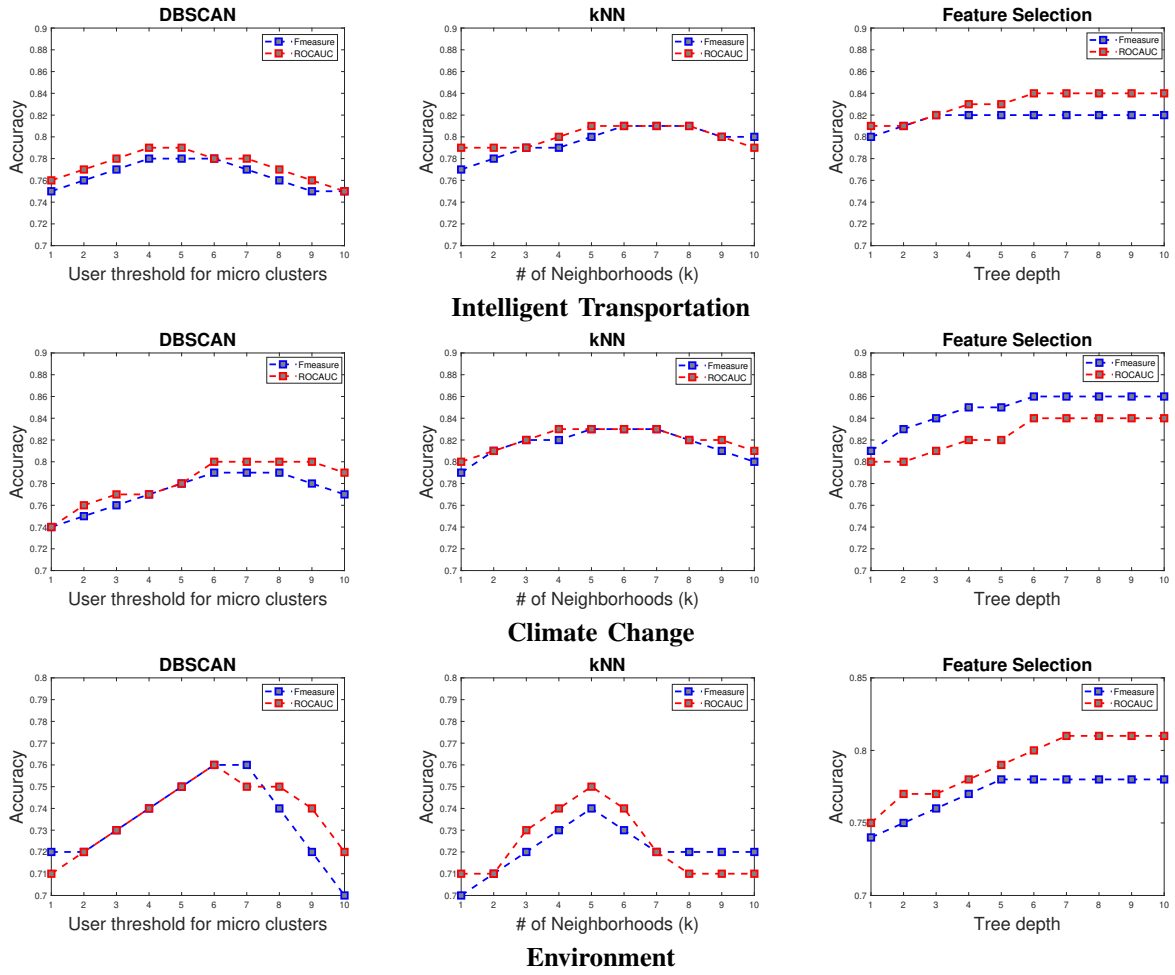

Fig. 2. The parameter setting of the data mining-based solutions.

TABLE II
PARAMETER SETTING OF CNN-GTO

CNN Architecture	Number of Epochs	Intelligent Transportation	Climate Change	Environment
AlexNet	100	0.81	0.82	0.80
	1000	0.85	0.82	0.81
	10000	0.87	0.84	0.83
VGG16	100	0.82	0.84	0.82
	1000	0.88	0.85	0.83
	10000	0.89	0.86	0.85
VGG19	100	0.85	0.87	0.85
	1000	0.90	0.88	0.86
	10000	0.92	0.88	0.87

and number of epochs (from 100 to 10000) for CNN-GTO. Regardless of the trajectory database used as input (Intelligent Transportation, Climate Change or Environment), the accuracy of DBSCAN-GTO and kNN-GTO is increased with increasing the parameter values, until a specified optimal point, and then starts decreasing. Regarding FS-GTO, and CI-GTO, the accuracy increased with increasing the parameters values, until stabilization at a specified optimal point. In case of CNN-GTO, the accuracy increased by increasing with the number of epochs, and the complexity of the CNN architecture. Thus, for low number of epochs, and simple architectures like AlexNet, the accuracy do not exceed 0.82 for all cases, however for large number of epochs and complex architectures like VGG19, the accuracy reach 0.92. The best parameter values obtained in this step are used in the remaining of the experiments. The best values of the proposed solutions for different trajectory databases are:

1) Intelligent Transportation, user threshold is set to 4 for DBSCAN-GTO, k is set to 6 for kNN-GTO, tree depth is set to 6 for FS-GTO.
2) Climate Change, user threshold is set to 6 for DBSCAN-GTO, k is set to 5 for kNN-GTO, tree depth is set to 6 for FS-GTO.
3) Intelligent Transportation, user threshold is set to 6 for DBSCAN-GTO, k is set to 5 for kNN-GTO, tree depth is set to 7 for FS-GTO.

For all trajectory databases (Intelligent Transportation, Climate Change, and Environment), the population size is set to 50, and the maximum number of iterations is set to 100 for CI-GTO, CNN architecture is set to VGG19, and number of epochs is set to 10000 for CNN-GTO. Moreover, several experiments have been carried out to tune the parameters MinPts, and ϵ for DBSCAN-GTO, and the distance measures for kNN-GTO. The results reveal that these parameters are not critical for the group outlier detection performance. Moreover, the density threshold has been varied from 0.1 to 1.0 for all

TABLE III
DENSITY THRESHOLD SETTING

Data	Algorithms	Density Threshold
Intelligent Transportation	DBSCAN-GTO	0.50
	kNN-GTO	0.40
	FS-GTO	0.45
	CI-GTO	0.64
	CNN-GTO	0.62
Climate Change	DBSCAN-GTO	0.47
	kNN-GTO	0.42
	FS-GTO	0.43
	CI-GTO	0.54
	CNN-GTO	0.69
Environment	DBSCAN-GTO	0.72
	kNN-GTO	0.76
	FS-GTO	0.84
	CI-GTO	0.91
	CNN-GTO	0.51

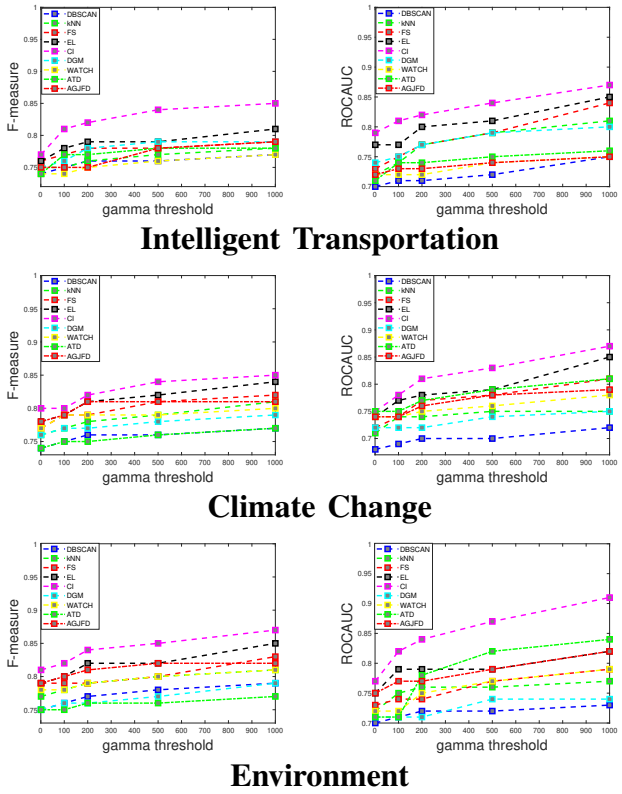

Fig. 3. Accuracy: The Proposed Solutions Vs. State-of-the art Traditional Group Detection.

algorithms. The best results are given in Table III.

B. Proposed Solutions Vs State-of-the-art Traditional Group Detection Solutions

The aim of this experiment is to compare the proposed solutions with the state-of-the art algorithms in terms of accuracy and processing time. To the best of our knowledge, this is the first work that explores group trajectory outlier detection. We therefore compare the proposed solutions with general group outlier detection solutions (see Section II). We adapted four well-known algorithms to trajectory data, i.e., DGM [5], WATCH [40], ATD [44], and AGJFD [45]. Fig. 3 and Fig. 4 present both accuracy and runtime of the proposed solutions (DBSCAN, kNN, Feature Selection, Ensemble Learning,

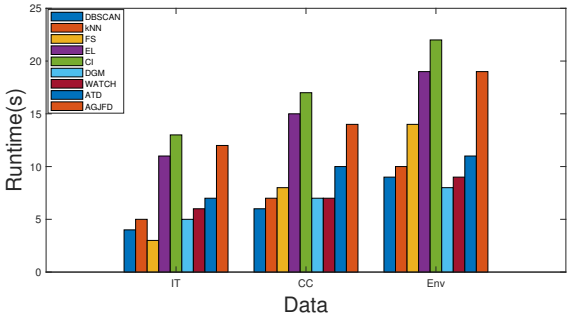

Fig. 4. Runtime: The Proposed Solutions Vs. State-of-the art Traditional Group Detection.

TABLE IV

Data	Algorithms	CPU	Fmeasure	ROCAUC
Taxi 13-1	DBSCAN-GTO	324	0.75	0.72
	kNN-GTO	337	0.76	0.71
	FS-GTO	345	0.78	0.74
	CI-GTO	371	0.83	0.79
	CNN-GTO	329	**0.89**	**0.88**
	GM-VSAE	331	0.84	0.81
	IRL-AD	**313**	0.82	0.82
	OCC	325	0.83	0.83
Taxi 13-2	DBSCAN-GTO	401	0.76	0.72
	kNN-GTO	429	0.77	0.72
	FS-GTO	475	0.80	0.77
	CI-GTO	483	0.86	0.81
	CNN-GTO	409	**0.92**	**0.90**
	GM-VSAE	**395**	0.85	0.82
	IRL-AD	415	0.84	0.83
	OCC	423	0.85	0.84

and Computational Intelligence), and the baseline algorithms (DGM, WATCH, ATD, and AGJFD). Note that the ensemble learning strategy merge the results of three data mining based algorithms. For each group of trajectory outliers, the number of the occurrences of the three algorithms (DBSCAN, kNN, and Feature Selection) is determined, while considering the groups that are highly frequent. For any gamma threshold from 1 to 1000, the solutions based on feature selection, ensemble learning, and computational intelligence methods outperform the baseline algorithms in terms of accuracy. However, solutions based on neighborhood computation and DBSCAN are less competitive. This comes from the fact that the former solutions use more advanced and recent strategies, while the latter solutions use less advanced concepts. Regarding the time processing, the advanced solutions require more time than the less advanced ones, but they are still competitive in this respect to the baseline algorithms.

C. Proposed Solutions vs. State-of-the-art Advanced Group Detection Solutions

In this experiment, the proposed solutions are compared with more advanced outlier detection algorithms on big trajectory databases. Three deep learning models for anomaly detection are chosen: Gaussian Mixture Variational Sequence AutoEncoder (GM-VSAE) [58], inverse reinforcement learning for Anomaly Detection (IRL-AD) [59], and One-class classification (OCC) [60]. The results are reported in Tab. IV, which reveal that CNN-GTO is very competitive compared to the

advanced outlier detection algorithms. It outperforms them terms of accuracy (both Fmeasure and ROCAUC values). However, it requires more computational time to deal with both trajectory databases. This result confirms the usefulness of deep learning to find the group of trajectory outliers. Nevertheless, the computational time should be investigated for reducing the number of bounding boxes.

D. Statistical Analysis and Discussions

The results obtained in the previous section are analyzed in the following through a Z-test statistical test . This includes twelve group trajectory outlier detection algorithms that have been used in this study; DBSCAN-GTO, kNN-GTO, FS-GTO, CI-GTO, and CNN-GTO, DGM, WATCH, ATD, AGJFD, GM-VSAE, IRL-AD, and OCC, along with the trajectory databases used in the experiments. The model is defined as follows: 1) Each algorithm is viewed as a normal variable. 2) Trajectory databases are divided into 100 partitions, and every partition contains 10% of the whole data. Each partition represents an observation, which generates 100 different observations, and 3) The result of every partition is considered as a sample.

Thirty three estimators (from E_1 to E_{33}) are used in the analysis. The first eleven estimators are designated for the runtime performance, the second eleven estimators are designated for the F-measure performance, and the last eleven estimators are designated for the AUC performance. The detailed description of these estimators are given as follows:

Performance estimators
E_1=CPU(DBSCAN-GTO)-CPU(kNN-GTO)
E_2=E_1-CPU(FS-GTO)
E_3=E_2-CPU(CI-GTO)
E_4=E_3-CPU(CNN-GTO)
E_5=E_4-CPU(DGM)
E_6=E_5-CPU(WATCH)
E_7=E_6-CPU(ATD)
E_8=E_7-CPU(AGJFD)
E_9=E_8-CPU(GM-VSAE)
E_{10}=E_9-CPU(IRL-AD)
E_{11}=E_{10}-CPU(OCC)
F-measure estimators
E_{12}=F(DBSCAN-GTO)-F(kNN-GTO)
E_{13}=E_{12}-F(FS-GTO)
E_{14}=E_{13}-F(CI-GTO)
E_{15}=E_{14}-F(CNN-GTO)
E_{16}=E_{15}-F(DGM)
E_{17}=E_{16}-F(WATCH)
E_{18}=E_{17}-F(ATD)
E_{19}=E_{18}-F(AGJFD)
E_{20}=E_{19}-F(GM-VSAE)
E_{21}=E_{20}-F(IRL-AD)
E_{22}=E_{21}-F(OCC)
AUC estimators
E_{23}=AUC(DBSCAN-GTO)-AUC(kNN-GTO)
E_{24}=E_{23}-AUC(FS-GTO)
E_{25}=E_{24}-AUC(CI-GTO)
E_{26}=E_{25}-AUC(CNN-GTO)
E_{27}=E_{26}-AUC(DGM)
E_{28}=E_{27}-AUC(WATCH)
E_{29}=E_{28}-AUC(ATD)
E_{30}=E_{29}-AUC(AGJFD)
E_{31}=E_{30}-AUC(GM-VSAE)
E_{32}=E_{31}-AUC(IRL-AD)
E_{33}=E_{32}-AUC(OCC)

where,
$CPU(A)$: is the average of the runtime values of the given algorithm A in the 100 observations.
$F(A)$: is the average of the F-measure values of the given algorithm A in the 100 observations.
$AUC(A)$: is the average of the AUC values of the given algorithm A in the 100 observations.
A: Algorithm belongs to the set {DBSCAN-GTO, kNN-GTO, FS-GTO, CI-GTO, and CNN-GTO, DGM, WATCH, ATD, AGJFD, GM-VSAE, IRL-AD, and OCC}.
First, the normality of the nine algorithms is checked using the Shapiro-Wilk test which is available on XLSTAT tool. Therefore, the first hypothesis, H_0, and the alternative hypothesis, H_α are defined as:
H_0: The algorithms follow a normal distribution.
H_α: The algorithms do not follow a normal distribution.

The used significance level (α) was set to 2%. The results of the Shapiro-Wilk test, indicated that H_0 cannot be rejected. Hence, the algorithms follow the normal distribution. The Z-test is then used with $\alpha = 2\%$ to compare the algorithms. XLSTAT shows E_{11}, E_{16}, and E_{27} give high values than the other estimators, which means that CNN-GTO is statistically better than the other algorithms in terms of F-measure and AUC, while IRL-AD is better than the other algorithms in terms of runtime.
From this analysis we can conclude that the deep learning based solution gives better performances compared to the data mining and computational intelligence based solutions. The deep learning based solution extract the relevant features using the convolution neural network. Several convolution and Max-Pooling layers are used to efficiently determine such features. Furthermore, accurate and fast RCNN are used in CNN-GTO to efficiently identify the group of trajectory outliers. However, pruning strategies may be investigated to reduce the number of bounding boxes, and therefore to improve the computational time of CNN-GTO.

IX. Conclusions and Future Work

The problem of Group Trajectory Outlier detection has been introduced in this paper, which consists in discovering group of trajectory outliers. This is different from previous trajectory outlier detection approaches that are only able to derive individual trajectory outliers. The GTO problem is particularly relevant to smart city applications, where a large volume of data on trajectories is collected daily. We propose three algorithms, *DBSCAN-GTO*, kNN-GTO, *FS-GTO*, and implementation on GPU. All approaches have been tested on real trajectory databases in comparison with baseline group detection algorithms, and the results demonstrate the usefulness of exploring ensemble learning, computational intelligence, and HPC in identifying group trajectory outliers. Moreover, the GPU-parallel approach outperforms the existing HPC approaches for dealing with big trajectory databases.

An interesting finding of this study is that the efficient combination of several concepts from different fields in detecting group of trajectory outliers improves the overall performance

(for both quality and runtime) compared to the baseline approaches. This result has been obtaoned by exploring machine learning algorithms (micro clusters, nearest neighbors, feature selection), as well as ensemble learning, computational intelligence and HPC. From a machine learning research standpoint, our GTO solutions are examples of the adaptation of generic algorithms to a specific context of trajectory outliers. As in many other cases, porting a pure machine learning technique into a specific application domain requires methodological refinement and adaptation [10], [44]. This work is only the first milestone for the use of ML in GTO, and much more investigation are needed. This research has potential to pave the way towards mature solutions that could be exploited by city planners in smart city environments.

The directions of future work include i) *Techniques for GTO*: more sophisticated techniques can be developed for *GTO* problem. For instance, other traditional outlier detection techniques may be adopted such as Local Outlier Factor (LOF) [61]. This can be done by developing new concepts of density and local reachability density for the *GTO* problem. ii) *Visualization*: new visualization techniques can be developed for *GTO*. This will provide accessible presentation of groups of trajectory outliers to the city planners. iii) *GTO applications*: more effort is needed for targeting new applications of *GTO*, such as climate change analysis, finding a group of hurricane trajectories that deviates from the normal hurricanes. This allows to identify other cities that could be affected by the hurricanes. The last hurricanes observed in the *United States* during the period of 2018-2019 is a typical example of a case study.

ACKNOWLEDGEMENT

This paper was supported by the Polish National Science Centre grant 2016/23/B/ST6/03599.

REFERENCES

[1] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[3] Y. Zhang, N. Meratnia, and P. J. Havinga, "Outlier detection techniques for wireless sensor networks: A survey." *IEEE Communications Surveys and Tutorials*, vol. 12, no. 2, pp. 159–170, 2010.

[4] S. Li, M. Shao, and Y. Fu, "Multi-view low-rank analysis with applications to outlier detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 12, no. 3, p. 32, 2018.

[5] R. Chalapathy, E. Toth, and S. Chawla, "Group anomaly detection using deep generative models," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 173–189.

[6] L. Xiong, B. Póczos, and J. G. Schneider, "Group anomaly detection using flexible genre models," in *Advances in neural information processing systems*, 2011, pp. 1071–1079.

[7] T. Harris, "A kohonen som based, machine health monitoring system which enables diagnosis of faults not seen in the training set," in *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, vol. 1. IEEE, 1993, pp. 947–950.

[8] R. Kannan, H. Woo, C. C. Aggarwal, and H. Park, "Outlier detection for text data," in *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 2017, pp. 489–497.

[9] J. Mao, P. Sun, C. Jin, and A. Zhou, "Outlier detection over distributed trajectory streams," in *Proceedings of the 2018 SIAM international conference on data mining*. SIAM, 2018, pp. 64–72.

[10] W. Li, H. Song, and F. Zeng, "Policy-based secure and trustworthy sensing for internet of things in smart cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 716–723, 2018.

[11] C. W. Landsea and J. L. Franklin, "Atlantic hurricane database uncertainty and presentation of a new database format," *Monthly Weather Review*, vol. 141, no. 10, pp. 3576–3592, 2013.

[12] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of KDD*, 1996, pp. 226–231.

[13] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *ACM SIGMOD Record*, vol. 29, no. 2, 2000, pp. 427–438.

[14] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework," in *Proc. of ICDE*, 2008, pp. 140–149.

[15] I. R. Pulshashi, H. Bae, H. Choi, and S. Mun, "Smoothing of trajectory data recorded in harsh environments and detection of outlying trajectories," in *Proceedings of the 7th International Conference on Emerging Databases*. Springer, 2018, pp. 89–98.

[16] M. A. Saleem, W. Nawaz, Y.-K. Lee, and S. Lee, "Road segment partitioning towards anomalous trajectory detection for surveillance applications," in *2013 IEEE 14th International Conference on Information Reuse & Integration (IRI)*. IEEE, 2013, pp. 610–617.

[17] I. San Román, I. M. de Diego, C. Conde, and E. Cabello, "Outlier trajectory detection through a context-aware distance," *Pattern Analysis and Applications*, vol. 22, no. 3, pp. 831–839, 2019.

[18] Y. Wang, K. Qin, Y. Chen, and P. Zhao, "Detecting anomalous trajectories and behavior patterns using hierarchical clustering from taxi gps data," *ISPRS international journal of geo-information*, vol. 7, no. 1, p. 25, 2018.

[19] J. Zhu, W. Jiang, A. Liu, G. Liu, and L. Zhao, "Time-dependent popular routes based trajectory outlier detection," in *International Conference on Web Information Systems Engineering*. Springer, 2015, pp. 16–30.

[20] Y. Ge, H. Xiong, Z.-h. Zhou, H. Ozdemir, J. Yu, and K. C. Lee, "Top-eye: Top-k evolving trajectory outlier detection," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, 2010, pp. 1733–1736.

[21] X. Kong, X. Song, F. Xia, H. Guo, J. Wang, and A. Tolba, "Lotad: Long-term traffic anomaly detection based on crowdsourced bus trajectory data," *World Wide Web*, vol. 21, no. 3, pp. 825–847, 2018.

[22] J. Mao, T. Wang, C. Jin, and A. Zhou, "Feature grouping-based outlier detection upon streaming trajectories," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2696–2709, 2017.

[23] J. Oehling and D. J. Barry, "Using machine learning methods in airline flight data monitoring to generate new operational safety knowledge from existing data," *Safety science*, vol. 114, pp. 89–104, 2019.

[24] Y. Yu, L. Cao, E. A. Rundensteiner, and Q. Wang, "Detecting moving object outliers in massive-scale trajectory streams," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 422–431.

[25] C. L. Yu, Yanwei, E. A. Rundensteiner, and Q. Wang, "Outlier detection over massive-scale trajectory streams," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 2, p. 10, 2017.

[26] Z. Zhu, D. Yao, J. Huang, H. Li, and J. Bi, "Sub-trajectory-and trajectory-neighbor-based outlier detection over trajectory streams," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2018, pp. 551–563.

[27] Q. Yu, Y. Luo, C. Chen, and X. Wang, "Trajectory outlier detection approach based on common slices sub-sequence," *Applied Intelligence*, vol. 48, no. 9, pp. 2661–2680, 2018.

[28] P. Banerjee, P. Yawalkar, and S. Ranu, "Mantra: a scalable approach to mining temporally anomalous sub-trajectories," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1415–1424.

[29] C. Chen, D. Zhang, P. S. Castro, N. Li, L. Sun, S. Li, and Z. Wang, "iboat: Isolation-based online anomalous trajectory detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 806–818, 2013.

[30] P.-R. Lei, "A framework for anomaly detection in maritime trajectory behavior," *Knowledge and Information Systems*, vol. 47, no. 1, pp. 189–214, 2016.

[31] Q. Lin, D. Zhang, K. Connelly, H. Ni, Z. Yu, and X. Zhou, "Disorientation detection by mining gps trajectories for cognitively-impaired elders," *Pervasive and mobile computing*, vol. 19, pp. 71–85, 2015.

[32] D. Zhang, N. Li, Z.-H. Zhou, C. Chen, L. Sun, and S. Li, "ibat: detecting anomalous taxi trajectories from gps traces," in *Proceedings of the 13th international conference on Ubiquitous computing*, 2011, pp. 99–108.

[33] S. Ando, T. Thanomphongphan, Y. Seki, and E. Suzuki, "Ensemble anomaly detection from multi-resolution trajectory features," *Data mining and knowledge discovery*, vol. 29, no. 1, pp. 39–83, 2015.

[34] W. Chu, H. Xue, C. Yao, and D. Cai, "Sparse coding guided spatiotemporal feature learning for abnormal event detection in large videos," *IEEE Transactions on Multimedia*, vol. 21, no. 1, pp. 246–255, 2018.

[35] S. Liu, L. Chen, and L. M. Ni, "Anomaly detection from incomplete data," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 9, no. 2, pp. 1–22, 2014.

[36] C. Piciarelli, C. Micheloni, and G. L. Foresti, "Trajectory-based anomalous event detection," *IEEE Transactions on Circuits and Systems for video Technology*, vol. 18, no. 11, pp. 1544–1554, 2008.

[37] H. Wu, W. Sun, and B. Zheng, "A fast trajectory outlier detection approach via driving behavior modeling," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 837–846.

[38] K. Das, J. Schneider, and D. B. Neill, "Anomaly pattern detection in categorical datasets," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 169–176.

[39] G. Tang, J. Pei, J. Bailey, and G. Dong, "Mining multidimensional contextual outliers from categorical relational data," *Intelligent Data Analysis*, vol. 19, no. 5, pp. 1171–1192, 2015.

[40] J. Li, J. Zhang, N. Pang, and X. Qin, "Weighted outlier detection of high-dimensional categorical data using feature grouping," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, no. 99, pp. 1–14, 2018.

[41] X. Zhao, J. Zhang, X. Qin, J. Cai, and Y. Ma, "Parallel mining of contextual outlier using sparse subspace," *Expert Systems with Applications*, vol. 126, pp. 158–170, 2019.

[42] L. Xiong, B. Póczos, J. Schneider, A. Connolly, and J. VanderPlas, "Hierarchical probabilistic models for group anomaly detection," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 789–797.

[43] R. Yu, X. He, and Y. Liu, "Glad: group anomaly detection in social media analysis," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 2, p. 18, 2015.

[44] H. Soleimani and D. J. Miller, "Atd: Anomalous topic discovery in high dimensional discrete data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 9, pp. 2267–2280, 2016.

[45] C. Sun, Z. Yan, Q. Li, Y. Zheng, X. Lu, and L. Cui, "Abnormal group-based joint medical fraud detection," *IEEE Access*, vol. 7, pp. 13 589–13 596, 2019.

[46] N. Modani and K. Dey, "Large maximal cliques enumeration in sparse graphs," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 1377–1378.

[47] J. D. Eblen, C. A. Phillips, G. L. Rogers, and M. A. Langston, "The maximum clique enumeration problem: algorithms, applications, and implementations," in *BMC bioinformatics*, vol. 13, no. 10. BioMed Central, 2012, p. S5.

[48] Y. Djenouri, A. Belhadi, J. C.-W. Lin, D. Djenouri, and A. Cano, "A survey on urban traffic anomalies detection algorithms," *IEEE Access*, vol. 7, pp. 12 192–12 205, 2019.

[49] Y. Li, Q. Huang, M. Kerber, L. Zhang, and L. Guibas, "Large-scale joint map matching of gps traces," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 214–223.

[50] C. Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet, "Online map-matching based on hidden markov model for real-time traffic sensing applications," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2012, pp. 776–781.

[51] M. A. Brubaker, A. Geiger, and R. Urtasun, "Map-based probabilistic visual self-localization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 4, pp. 652–665, 2015.

[52] A. Prokhorchuk, J. Dauwels, and P. Jaillet, "Estimating travel time distributions by bayesian network inference," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[53] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: a partition-and-group framework," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, 2007, pp. 593–604.

[54] H. Li, J. Liu, K. Wu, Z. Yang, R. W. Liu, and N. Xiong, "Spatio-temporal vessel trajectory clustering based on data mapping and density," *IEEE Access*, 2018.

[55] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[56] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[57] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 5, pp. 649–659, 2008.

[58] Y. Liu, K. Zhao, G. Cong, and Z. Bao, "Online anomalous trajectory detection with deep generative sequence modeling," *ICDE. IEEE*, 2020.

[59] M.-h. Oh and G. Iyengar, "Sequential anomaly detection using inverse reinforcement learning," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1480–1490.

[60] M. Sabokrou, M. Fathy, G. Zhao, and E. Adeli, "Deep end-to-end one-class classifier," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[61] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM SIGMOD Record*, vol. 29, no. 2, 2000, pp. 93–104.

Asma Belhadi received PhD in Computer Science at USTHB in 2016. Dr Asma Belhadi is currently a postdoctoral researcher at Kristiania College University. She published more than 30 papers in the areas of artificial intelligence, and data mining.

Youcef Djenouri obtained the PhD in Computer Science from USTHB Algiers, Algeria, in 2014. He is currently a research scientist at SINTEF Digital, Oslo, Norway. Dr Youcef Djenouri has published more than 60 refereed research papers, in the areas of data mining, parallel computing and artificial intelligence.

Djamel Djenouri obtained the PhD in Computer Science from USTHB, Algiers, Algeria, in 2007. He is currently an Associate Professor at the University of the West England in Bristol, UK. He is working on topics related Internet of things, wireless and mobile networks, machine learning and application for smart cities and green applications. He published more than 100 papers, two books, and two national patents.

Tomasz Michalak is an Associate Professor at University of Warsaw. From June 2012 to May 2017, he was also a part-time Post-Doctoral Researcher at the Department of Computer Science, University of Oxford, UK. His research interest focus on artificial intelligence, and machine learning applications.

Jerry Chun-Wei Lin received the Ph.D. in Computer Science and Information Engineering from National Cheng Kung University, Tainan, Taiwan, in 2010. He is now working as an associate professor at HVL, Bergen, Norway. His research interests include data mining, privacy-preserving and security, Big Data analytics, and social networks. He has published more than 200 research papers in peer-reviewed international conferences and journals.